Subspace-based Algorithms for Multivariable System Identification

Vasile Sima
Computer Process Control Laboratory
Research Institute for Informatics
8–10 Averescu Avenue,
71316 Bucharest 1
ROMANIA

Abstract: Basic algorithms for multivariable system identification by subspace techniques are briefly described. Deterministic and combined deterministic-stochastic identification problems are dealt with using two approaches. A state space model is computed from input-output data sequences. Multiple data sequences, collected by possibly independent identification experiments, can be handled. Sequential processing of large data sets is provided as an option. Illustrative numerical examples are included.

Keywords: control system design; identification methods; invariant subspaces; least squares solutions; multivariable systems, numerical linear algebra; QR factorization; singular value decomposition.

Dr. Vasile Sima was born at Lita, Romania, on the 21st of October, 1949. He graduated from the Polytechnical Institute of Bucharest in Control Engineering in 1972, and from the Department of Mathematics at the University of Bucharest, in 1978. He obtained his doctoral degree in Control Engineering (adaptive control) from the Polytechnical Institute of Bucharest, in 1983. Since 1972 he has held several research positions at the Research Institute for Informatics in Bucharest. He is senior research worker and vicechairman of the Scientific Board of the institute. He is also an Associate Professor at the "Politehnica" University of Bucharest. Dr. Vasile Sima has published more than 80 scientific papers (about 35 of them were published in international journals and symposia proceedings). He co-authored two books (in Romanian): "Computer-Aided Optimization Practice" and "Adaptive and Flexible Control of Industrial Processes", and authored other two books "New Methods in Applied Mathematics" (in Romanian), and "Algorithms for Linear-Quadratic Optimization" (Marcel Dekker, Inc., New York). He is a member of NA-NET (Numerical Analysis Network), WGS (Working Group on Software), and an affiliate member of IFAC (International Federation of Automatic Control). His research interests include automatic control theory, adaptive and optimal control, computer-aided control systems design, nonlinear programming, numerical linear algebra and scientific computations.

1. Introduction

Computer-aided design of control systems is usually based on linear time-invariant (LTI) state space models. Surprisingly enough, multivariable state space system identification has only recently become a topic of intense research. In contrast with classical input-output (I/O) identification approaches, the newly proposed Subspace Model Identification (SMI) techniques essentially find a state sequence, or a column space approximation, and then determine the system matrices by solving some least-squares problems. These techniques have promising advantages over the classical ones. One advantage is that there is no need for parameterizations, which are notoriously difficult to choose or analyze, or could lead to ill-conditioned problems. Some other advantage will be that robust numerical linear algebra techniques, like QR factorization and singular value decomposition (SVD), can largely be applied; this is in contrast with the iterative optimization schemes required in the parametric model identification approach, documented e.g. in [1]. The attractiveness of SMI techniques is even more exercised by the small number of parameters (essentially only one) to be selected for determining the model structure, without any restriction on model generality. See, for instance, [2]–[7] for a further discussion of the SMI features.

MATLAB codes based on SMI algorithms have lately been developed, e.g. in [8]. For efficiency and accuracy reasons, it is no doubt useful to implement some algorithms in Fortran, using the state-of-the-art, public-domain linear algebra package LAPACK [9]. This allows to exploit the potential parallelism of many modern computer architectures.

This paper briefly describes some basic algorithms, implemented in Fortran, for discrete-time multivariable system identification by subspace techniques. Two classes of SMI techniques to identify an LTI system are considered. The first one is referred to as the State Intersection (SI) class of SMI techniques. The implemented method, described in [2], is known as the N4SID (Numerical algorithm for Subspace State Space System IDentification) approach. The second one is referred to as the Multivariable Output Error state SPace (MOESP) class of techniques. The implemented MOESP methods are described in [3], [5], and [6]. Deterministic and combined deterministic-stochastic identification problems are dealt with. A state space model is computed from I/O data sequences. It is possible to handle multiple I/O sequences, each one being collected during a possibly independent identification experiment. Sequential processing of large data sets is optional. In addition to the main identification facilities, several auxiliary routines are available to perform kernel linear algebra computations, or solve related identification problems: identification of a limited set of Markov parameters, shifting the I/O data sequences to reduce the state dimension when there are dead-times in different input channels, estimation of the initial conditions, identification of systems operating in closed loop, or simulation of discrete-time LTI systems.

A theoretical statement of an algorithm and its efficient and reliable implementation are quite apart. In implementing algorithms, great attention was paid to developing new LAPACK-style codes for special QR or singular value decompositions, to exploit the particular structure of the problem, to increase efficiency, and reduce the memory requirements. Theoretical algorithms and their MATLAB realizations have been largely reorganized. For instance, the N4SID algorithm makes use of the inverse of a part of the triangular factor, L, of an LQ factorization of a Hankel-like matrix constructed from I/O sequences. (It should precede the computing of the SVD that gives the system order and then the quadruple of system matrices.) The MATLAB implementations solve several standard least-squares problems to obtain the so-called associated "oblique projection". The new LAPACK-based implementation only involves some orthogonal transformations for obtaining various projections and residuals. It is worth-mentioning that the MOESP approach is apparently more attractive from a numerical point of view than the N4SID approach is, because of its performing the singular value decomposition just on a submatrix of the above-mentioned triangular factor L. As previously specified, the literature makes use of an LQ factorization of the Hankel-like matrix. While convenient from a theoretical point of view, in practice the QR factorization is to be preferred in computations. For instance, there is no code available for performing the LQ factorization with pivoting, and pivoting is needed for certain calculations where a rank decision should be made. (For example, the covariance calculations in the N4SID approach require such a decision.) The current MATLAB codes compute the lower triangular factor $L = R^T$, then determine the singular value decomposition of R^T, $R^T = U\Sigma V^T$, and use the singular values in Σ for finding the system order, n (by visually detecting the widest "gap" between two consecutive singular values), and the left singular vectors in U for determining system matrices. Our approach consists in computing $R = U\Sigma V^T$, and using the right singular vectors (columns in V^T).

One objective of the Fortran implementation has been to call the LAPACK routines as much as possible. Preference has generally been for the block variants and the Level III BLAS codes, which are responsible for efficient use of parallelism. In order to reduce the object code volume, we have not called some very large LAPACK routines, like DGESVD. Whenever possible, the problem structure has been exploited. For instance, the calculations have been organized such that singular value decompositions are required for triangular matrices only. A dedicated, very compact and efficient routine has been purposely written. Moreover, in both theory and the MATLAB implementations, the computation of the system matrices involves a very large matrix. Fortunately, this matrix can be brought to a special block-triangular structure. A structure-exploiting QR row compression scheme has been devised, reducing the storage requirements and the computational effort.

Some other related objective has been that of enabling the performance of computations with as reduced memory requirements as possible, but without sacrificing the speed, when enough memory is available. For instance, there is an option for sequential calculation of the triangular factor in the QR factorization of the Hankel-like matrix. The user can specify that I/O data are accessed in (independent or not) batches. Moreover, when there is not enough working memory for processing a given data batch, the codes automatically operate an inner sequential processing of that batch, to accommodate with the available memory space. (A well-motivated lower limit of the memory space is however required.) By minimizing the memory access, processing of large I/O sequences and identification of systems of large order become possible, even on computers with reduced memory resources. No provisions are made for using the displacement rank techniques (based on the block-Hankel structure) [10] in current implementations. Whereas such techniques could improve efficiency, they are less stable numerically.

Stochastic part identification is utterly more difficult than the deterministic part identification. The determination of the system order n is even much subtler, because there is no clear "gap" in the singular values. For the time being, no "best" solution of stochastic identification problem does exist and many details of the algorithms proposed in the literature are based on heuristic results. Some algorithms are labelled as "robust" [7], in the sense that they have demonstrated a good behaviour on many sets of industrial data.

Another important research topic is related with preserving the positivity of the full covariance matrix; this is not guaranteed by the usual algorithms. The positivity is equivalent to the real positivity of a certain covariance sequence, or to the strict positivity of the solutions to associated Riccati equations. The positivity condition for such covariance sequence is rarely (and hardly) satisfied, even for data generated by simulation of a linear stochastic system. This is due to either a finite number of available data samples, or the fact that the real data are not truly generated by an LTI stochastic system.

The identification algorithms can be modified to preserve positivity. Specifically, after having determined matrices A, and C (and, eventually, B and D), a global covariance matrix is computed from residuals, a Lyapunov equation is solved, and its solution is used to define a Riccati equation, and finally an innovation model is obtained. If the state matrix A is stable, then the Lyapunov equation solution is positive definite. The only unpleasant aspect in preserving positivity is that only biased, not consistent, estimates are obtained.

The performances of the investigated algorithms in solving some simulated or real-world problems are under evaluation. The results obtained by using various algorithms (including their MATLAB versions) have been sharply compared. This has determined better confidence in the implemented algorithms; moreover, the analysis of the casually observed differences between the results yielded by different algorithms revealed several bugs in some MATLAB codes and in the corresponding initially written Fortran codes. The comparisons supposed some modifications in the original MATLAB codes, as to use the same basis for the state space representation (for instance, suppressing the scaling of the first n singular vectors—producing A and C matrices—by the reciprocals of the corresponding singular values). The analysis also revealed an interesting fact: the matrices computed using the minimum memory option could have had elements with the same absolute values, but with opposite signs, compared to the large memory case. Two such representations are related by a similarity transformation.

One difficulty encountered in identifying some system models (even with simulated data) is described below. Consider a deterministic system with poles on the unit circle. (Clearly, unstable, but controllable systems can be identified.) In this case, the computation of the covariance matrices is not necessary. However, if the matrices are computed, and if the positivity-preserving device described above is used, the corresponding discrete-time Lyapunov equation has no solution, because its A matrix has eigenvalues of unit modulus. Ill-conditioned calculations do appear in such situations. This suggests that such devices should only be included as options. Alternately, there should be special code portions which should automatically select the appropriate computation, based on the given problem characteristics, and notify the user on whether his options are adequate or not for the problem concerned.

2. Basic Approaches

The basic LTI state space models considered are described by

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k, \\ y_k &= Cx_k + Du_k + v_k, \end{aligned} \qquad (1)$$

where x_k is the n-dimensional state vector at time k, u_k is the m-dimensional input (control) vector, y_k is the ℓ-dimensional output vector, $\{w_k\}$ and $\{v_k\}$ are state and output disturbance or noise sequences, and A, B, C, and D are real matrices of appropriate dimensions. The system order, n, and the quadruple of system matrices (A, B, C, D) are not known; the only available information is given by an upper bound, s, on n, and by the input and output data sequences, $\{u_k\}$ and $\{y_k\}$, $k = 1{:}t$ (i.e., for k taking values from 1 to a given t).

2.1. The SI Approach

The main feature of the SI class of SMI techniques is the determination of either the state sequence of the LTI system to be identified, or of an observer to reconstruct its state sequence, via the intersection of the row spaces of the Hankel-like matrices constructed from "past" and "future" I/O data. The basic idea was introduced in [4]. An extension of this idea led to the N4SID algorithm developed in [2]. This algorithm identifies LTI state space models in the so-called innovation form, described by

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Kv_k, \\ y_k &= Cx_k + Du_k + v_k, \end{aligned} \qquad (2)$$

where $\{v_k\}$ is a zero-mean white noise sequence and $\{u_k\}$ is a deterministic input sequence (perfectly known to the user). Both the basic and extended variants of this class produce statistically consistent and efficient estimates when certain assumptions hold.

2.2. The MOESP Approach

The main feature of this class of SMI techniques is to determine an extended observability matrix of the deterministic part of the model (1). The extended observability matrix Γ_s is given by

$$\Gamma_s = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{s-1} \end{bmatrix}.$$

The basic idea was introduced in [5] and [6]. The simplest algorithm derived from this idea, called the ordinary MOESP scheme, allows to identify, in a statistically consistent and efficient way, LTI systems that can be described by (1), with $w_k \equiv 0$ and $\{v_k\}$ a zero-mean white noise sequence, independent of the input.

Extensions based on using past input quantities and/or on reconstructed state variables as instrumental variables have been proposed, allowing to consistently identify a model in (1), for $w_k \equiv 0$, and $\{v_k\}$ a zero-mean arbitrary stochastic disturbance, independent of the input. This increased applicability is made at the risk of not having efficient estimates. These variants are referred to as the PI or RS schemes, when past inputs or reconstructed state variables are used as instrumental variables, respectively.

When the additive disturbance w_k in (1) is generated by an innovation model of the form $w_k = Kv_k$, and v_k is a zero-mean white noise independent of the input, it is possible again to obtain both consistent and efficient estimates when, beside the past input, past output quantities are also used as instrumental variables. The scheme derived from this extension of ordinary MOESP scheme is known as the PO scheme [3].

For the deterministic-stochastic identification problem, it is possible to estimate by means of the PO scheme the deterministic part, and by means of the SI approach [2] the stochastic part, that is the noise covariance matrices. Further, the corresponding Kalman gain K can be computed to allow the design of state observers [3].

The algorithms entering the MOESP class have that striking feature of being highly streamlined, in the sense that the sequence of computations performed by these schemes is almost independent of the type of problems to be analyzed. This has permitted highly modular implementations of the algorithms under this class.

3. Basic Algorithms

The main algorithms are:

- The ordinary MOESP scheme, producing consistent and statistically efficient estimates of the state space quadruple (A, B, C, and D), of the deterministic part, for v zero-mean white noise, independent of the input.
- The ordinary MOESP scheme extended with instrumental variables based on reconstructed state and/or past input quantities, producing consistent estimates when v is zero-mean, but of arbitrary statistical color.
- The ordinary MOESP scheme extended with instrumental variables based on past input and output quantities, producing consistent and statistically efficient estimates when v is generated by an innovation model. Optionally, a Kalman predictor is also computed.
- The N4SID scheme, producing consistent and statistically efficient estimates when v is generated by an innovation model. Optionally, a Kalman predictor is also computed.

The MOESP scheme with past input and output quantities uses the N4SID approach for covariance calculations; this seems to be a very promising alternative to the pure N4SID approach. Full algorithmic details cannot be given here, due to space limitation. Only some key points will be illustrated.

3.1. Algorithmic Outline

The simplest algorithm corresponds to the ordinary MOESP scheme. For non-sequential data processing, the $N \times (m+\ell)s$ matrix $H = [\ U_{1,s,N}^T \quad Y_{1,s,N}^T\ ]$ is constructed, where N denotes the total number of samples that can be used (here, $N = t-s+1$), $U_{1,s,N}$ and $Y_{1,s,N}$ are block-Hankel matrices defined in terms of the input and output data, respectively, i.e.,

$$U_{1,p,N} = \begin{bmatrix} u_1 & u_2 & u_3 & \cdots & u_N \\ u_2 & u_3 & u_4 & \cdots & u_{N+1} \\ u_3 & u_4 & u_5 & \cdots & u_{N+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_p & u_{p+1} & u_{p+2} & \cdots & u_{N+p-1} \end{bmatrix},$$

and similarly for Y. A QR factorization $H = QR$, is used for data compression; then, a SVD of the submatrix $R_{22} := R(ms+1\colon(m+\ell)s, ms+1\colon(m+\ell)s)$ reveals the order n of the system as the number of "non-zero" singular values. System matrices are finally computed from the right singular vectors of R_{22}, using the $ms \times (m+\ell)s$ upper trapezoidal submatrix of R, and solving a linear algebraic system in a total least-squares sense [11] (see the next subsection).

For sequential data processing, the QR factorization is done sequentially, by updating the upper triangular factor R. Once all data compressed, the system order and system matrices are computed as in the previous case.

For other schemes, the things are similar, but more involved. For the PO scheme, R_{22} is given by

$$R_{22} := R(ms+1\colon(2m+\ell)s, ms+1\colon(2m+\ell)s).$$

For the N4SID scheme, the triangular factor R of the $N \times 2(m+\ell)s$ matrix $H = [\ U_{1,2s,N}^T \quad Y_{1,2s,N}^T\ ]$ is found ($N = t-2s+1$), an oblique projection O is computed in terms of some submatrices of the upper triangular factor R [2], and then a SVD of O gives the order n. System matrices are finally computed using the first n right singular vectors of O^T, and other submatrices of R, solving a linear algebraic system in a total least-squares sense. The covariance matrices are computed using the residuals of a least-squares problem. The Kalman gain is obtained by solving a discrete-time algebraic matrix Riccati equation based on the Schur vectors approach for the dual of an optimal control problem (see, for example, [12]).

To give further details, let us partition the triangular factor of H as $R = [U_p \;\; U_f \;\; Y_p \;\; Y_f]$, where the subscripts p and f stand for "past" and "future" data, respectively, and the four blocks have ms, ms, ℓs, and ℓs columns, respectively. Define $W_p = [U_p \;\; Y_p]$, and consider the residuals of the two least-squares problems giving the oblique projection,

$$r_1 = W_p - U_f X_1, \qquad r_2 = Y_f - U_f X_2,$$

where X_1 and X_2 are the minimum norm least-squares solutions of the following problems

$$\min \|U_f X - W_p\|_2, \qquad \min \|U_f X - Y_f\|_2,$$

respectively. Then, the oblique projection can be computed as $O = r_2^T Q_1 Q_1^T$, where Q_1 consists of the first $k = \operatorname{rank}(r_1)$ columns of the orthogonal matrix in the QR factorization of r_1. No least-squares problems should be actually solved.

3.2. Computation of System Matrices

The description below shows how the MOESP approach can estimate the quadruple of system matrices (A, B, C, D) of the LTI state space model using the information from previous computations.

Let R be a matrix whose leading $ms \times (m + \ell)s$ submatrix contains some relevant data for MOESP algorithm, namely the upper triangular matrix R_{11}, and the matrix R_{12}. (For the ordinary MOESP scheme, it is just the leading submatrix of the triangular factor of the QR factorization of H.) Let U be the $\ell s \times \ell s$ matrix of right singular vectors. (Here, U has been redefined as V^T in the corresponding singular value decomposition.)

The matrix C is readily obtained as the leading $\ell \times n$ submatrix of U. Denoting $U_1 = U(1\colon(s-1)\ell, 1\colon n)$, and $U_2 = U(\ell+1\colon \ell s, 1\colon n)$, the QR decomposition of U_1 is computed, and U_2 is updated accordingly,

$$U_1 = Q \begin{bmatrix} \widetilde{R} \\ 0 \end{bmatrix}, \qquad U_2 \leftarrow Q^T U_2.$$

Then, matrix A is obtained by solving $\widetilde{R}A \doteq U_2(1\colon n, :)$ for A.

Finding the B and D matrices is more involved. First, the matrix equation $K R_{11}^T = \widetilde{U}_2^T R_{12}^T$ is solved for K, where $\widetilde{U}_2 = U(:, n+1\colon \ell s)$, and K is an $(\ell s - n) \times ms$ matrix. The matrix K is stored in the workspace, **K**. Then, compute the triangular factor of the QR factorization of the $s(\ell s - n) \times (\ell s + m)$ structured matrix $[Q \;\; K_e]$, implicitly defined by

$$\begin{bmatrix} Q_{1s} & Q_{1,s-1} & \cdots & Q_{12} & Q_{11} & K_1 \\ 0 & Q_{1s} & \cdots & Q_{13} & Q_{12} & K_2 \\ 0 & 0 & \cdots & Q_{14} & Q_{13} & K_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & Q_{1s} & K_s \end{bmatrix},$$

where $Q_{1i} = \widetilde{U}_2(\ell(i-1)+1\colon \ell i, :)^T$ is $(\ell s - n) \times \ell$, for $i = 1\colon s$,

$$K_e = \begin{bmatrix} K_1^T & K_2^T & \cdots & K_s^T \end{bmatrix}^T,$$

and $K_i = K(:, (i-1)m+1\colon im)$ is $(\ell s - n) \times m$, $i = 1\colon s$. The matrix $[Q \;\; K_e]$ is triangularized in an array **R**, exploiting its structure (for efficient use of the memory space), so that on output, the upper triangle of **R** contains, in its leading $(\ell s+m) \times (\ell s+m)$ submatrix, the required triangular factor. The calculations need $0((\ell s+m)^3)$ floating point operations. They are briefly described below. The first block-row of Q, i.e. the $(\ell s - n) \times \ell s$ matrix $[Q_{1s} \;\; \cdots \;\; Q_{12} \;\; Q_{11}]$, is constructed in **R**. The submatrix Q_{1s} is triangularized using an orthogonal matrix S, and the transformation S^T is then applied to the matrix $[Q_{1,s-1} \;\; \cdots \;\; Q_{11}]$. Hence,

$$S^T [Q_{1s} \cdots Q_{11}] = \begin{bmatrix} R & P_{s-1} & \cdots & P_2 & P_1 \\ 0 & F_{s-1} & \cdots & F_2 & F_1 \end{bmatrix}.$$

The transformation S^T is also applied to each of the submatrices K_i of K_e, $i = 1\colon s$. Denote

$$[C_i^T \;\; G_i^T]^T = S^T K_i \quad (i = 1\colon s),$$

where C_i has ℓ rows. (Finally, C_i is saved in $\mathtt{R}(\ell(i-1)+1{:}\,\ell i, \ell s+1{:}\,\ell s+m)$, and G_i is in $\mathtt{K}(\ell+1{:}\,\ell s-n, m(i-1)+1{:}\,mi)$, $i=1{:}\,s$.) Then, the rows to be annihilated are put in $F(1{:}\,\ell s - n - \ell, 1{:}\,\ell s-\ell)$. (On implementation, F could overwrite $\widetilde{U}_2$.)

Now, the structure of the transformed matrix is:

$$\left[\begin{array}{ccccccc} R & P_{s-1} & P_{s-2} & \cdots & P_2 & P_1 & C_1 \\ 0 & R & P_{s-1} & \cdots & P_3 & P_2 & C_2 \\ 0 & 0 & R & \cdots & P_4 & P_3 & C_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & R & P_{s-1} & C_{s-1} \\ 0 & 0 & 0 & \cdots & 0 & R & C_s \\ 0 & F_{s-1} & F_{s-2} & \cdots & F_2 & F_1 & G_1 \\ 0 & 0 & F_{s-1} & \cdots & F_3 & F_2 & G_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & F_{s-1} & G_{s-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & G_s \end{array}\right],$$

where the block-rows have been permuted, to better exploit the structure. The block-rows having R on the diagonal are dealt with successively in the array $\mathtt{R}$. The F submatrices are stored in an auxiliary $(\ell s - n - \ell, \ell s - \ell)$ array $\mathtt{F}$, as a block-row. The large matrix above is never explicitly constructed. Only its first s block-rows are constructed, one by one. In order to triangularize the transformed matrix, exploiting its structure, part of the preceding block-row is first copied in the subsequent block-row, and then the current submatrix F_{s-i} is annihilated using an orthogonal matrix that modifies the corresponding submatrix R (in the same block-column). The transformation applies to the corresponding block-rows of the arrays $\mathtt{R}$, $\mathtt{F}$ and $\mathtt{K}$ (for G).

Finally, after having copied C_i, $i=1{:}\,s$, in the last block-column of the QR factor, $\mathtt{R}(1{:}\,\ell s, \ell s + 1{:}\,\ell s+m)$, $i=1{:}\,s$, the remaining rows of the transformed G are then compressed in the trailing part of $\mathtt{R}$,

$$\mathtt{R}(\ell s+1{:}\,\ell s+m, \ell s+1{:}\,\ell s+m).$$

Now, compute the right singular vectors matrix, V, of the triangular factor $\mathtt{R}$, of order $\ell s+m$. (V^T is stored in $\mathtt{R}$.) Solve for Y (the total least-squares solution) the matrix equation

$$V(\ell s+1{:}\,\ell s+m, \ell s+1{:}\,\ell s+m)^T Y^T = -V(\ell s+1{:}\,\ell s+m, 1{:}\,\ell s)^T,$$

where Y is $\ell s \times m$. Then, matrix D is readily obtained, if needed, as the trailing $\ell \times m$ submatrix of Y.

After re-arranging the block-rows of U_1 as follows

$$\left[\begin{array}{c} U_1(\ell(s-2)+1{:}\,\ell(s-1),:) \\ \vdots \\ U_1(\ell+1{:}\,2\ell,:) \\ U_1(1{:}\,\ell,:) \end{array}\right],$$

the modified U_1 matrix, extended by the first $\ell s - \ell$ rows of Y on the right, is triangularized. Finally, the right singular vectors matrix, V, of the triangular factor is computed, and the matrix equation

$$V(n+1{:}\,n+m, n+1{:}\,n+m)^T B^T = -V(n+1{:}\,n+m, 1{:}\,n)^T,$$

is solved for B (in a total least-squares acceptance).

For the N4SID algorithm, a similar computational scheme is used. In this case, the submatrices Q_{1i} are $(n+\ell)\times\ell$, for $i=1{:}\,s$, defined by

$$Q_{11} = \left[\begin{array}{c} -\mathcal{L}_{11} \\ I_\ell - \mathcal{L}_{21} \end{array}\right], \qquad i=1,$$

$$Q_{1i} = \left[\begin{array}{c} \mathcal{M}_{i-1} - \mathcal{L}_{1i} \\ -\mathcal{L}_{2i} \end{array}\right], \qquad i=2{:}\,s,$$

where $\mathcal{L}_{1i}$, $\mathcal{L}_{2i}$, and $\mathcal{M}_i$ are $n\times\ell$, $\ell\times\ell$, and $n\times\ell$, submatrices of $\mathcal{L}$, and $\mathcal{M}$, namely

$$\mathcal{L} = \left[\begin{array}{cccc} \mathcal{L}_{11} & \mathcal{L}_{12} & \cdots & \mathcal{L}_{1s} \\ \mathcal{L}_{21} & \mathcal{L}_{22} & \cdots & \mathcal{L}_{2s} \end{array}\right],$$

$$\mathcal{M} = \left[\begin{array}{cccc} \mathcal{M}_1 & \mathcal{M}_2 & \cdots & \mathcal{M}_{s-1} \end{array}\right],$$

and

$$\mathcal{L} := \left[\begin{array}{c} A \\ C \end{array}\right]\Gamma_s^\dagger, \qquad \mathcal{M} := \Gamma_{s-1}^\dagger,$$

the superscript $\dagger$ denoting the pseudoinverse. Note that here $\Gamma_s := U(:,1{:}\,n)$, and $\Gamma_{s-1} :=$

U_1. In this case, the submatrix K_e of the $s(n+\ell) \times (\ell s + m)$ structured matrix $[Q \;\; K_e]$, whose QR factorization should be computed, is defined by

$$K_e = [\; K_{11}^T \;\; K_{21}^T \;\; \cdots \;\; K_{1s}^T \;\; K_{2s}^T \;]^T ,$$

and matrix K is an $(n+\ell) \times ms$ matrix, given by

$$K = \begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1s} \\ K_{21} & K_{22} & \cdots & K_{2s} \end{bmatrix},$$

$K_{1i} = K(1\colon n, (i-1)m + 1\colon im)$ is $n \times m$, and $K_{2i} = K(n+1\colon n+\ell, (i-1)m+1\colon im)$ is $\ell \times m$, $i = 1\colon s$. It is assumed that matrices A, C, and K have already been computed, by solving a certain least-squares problem. The residuals of this problem are used to estimate the covariance matrix of the noise process $[w^T \;\; v^T]^T$, which helps in obtaining the Kalman predictor gain matrix. The calculations are then performed as in the MOESP case. Note that the row dimension of the structured matrix $[Q \;\; K_e]$ could be significantly less than with the MOESP case, so sequential processing little benefits at this stage. For small-scale systems, it would be more efficient to directly compute the QR factor of the matrix $[Q \;\; K_e]$.

4. Examples

First, consider the discrete-time system (A, B, C, D), due to M. Verhaegen, with the following matrices:

$$A = \begin{bmatrix} 1.5 & -0.7 \\ 1.0 & 0.0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0.5 \end{bmatrix},$$

and $D = 0$, whose output response $\{y_k\}$ is computed using a random (0,1) uniform distribution input trajectory $\{u_k\}$, $k = 1\colon 120$ ($t = 120$), and zero initial state. The value of parameter s is taken as 10.

All the calculations have been performed in double precision on an IBM PC 486 (with machine precision about 2.22×10^{-16}). The computed singular values used to estimate the system order, rounded to five significant digits, are: 32.203, 14.612, 0, 0, 0, 0, 0, 0, 0, 0. The estimated system matrices (rounded to four digits after the decimal point) are:

$$\widehat{A} = \begin{bmatrix} .7831 & .5606 \\ -.2472 & .7169 \end{bmatrix}, \quad \widehat{B} = \begin{bmatrix} -3.5468 \\ -2.3707 \end{bmatrix},$$

$$\widehat{C} = \begin{bmatrix} -.5749 & .4382 \end{bmatrix}, \quad \widehat{D} = 0.$$

The full precision computed matrices coincide with those obtained by applying an appropriate similarity transformation to the original system. (For such a simple example, this similarity transformation can easily be obtained. For more complex systems, a simple check on similarity is to compare the first two Markov parameters for the computed and the original system, namely to evaluate the absolute errors $||CB - \widehat{C}\widehat{B}||$, and $||CAB - \widehat{C}\widehat{A}\widehat{B}||$, or the corresponding relative errors.) The relative output root-mean-square error is:

$$\left(\sum_{k=1}^{t} (y_k - \widehat{y}_k)^2 \right)^{1/2} \Bigg/ \left(\sum_{k=1}^{t} y_k^2 \right)^{1/2} = 1.13 \times 10^{-15},$$

where $\widehat{y}_k$ is computed using the estimated quadruple ($\widehat{A}$, $\widehat{B}$, $\widehat{C}$, $\widehat{D}$), and the same input sequence and initial state as above.

As a second example, consider the following system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ew_k, \\ y_k &= Cx_k + Du_k + Fv_k, \end{aligned}$$

where [13]:

$$\begin{aligned} A \;\; = \;\; & \text{block-diag}\left(\begin{bmatrix} .9856 & .1628 \\ -.1628 & .9856 \end{bmatrix}, \right. \\ & \left. \begin{bmatrix} .8976 & .4305 \\ -.4305 & .8976 \end{bmatrix}, \begin{bmatrix} .8127 & .5690 \\ -.5690 & .8127 \end{bmatrix} \right), \end{aligned}$$

$$B = \begin{bmatrix} .0011 & .0134 & -.0016 & -.0072 & .0011 & .0034 \end{bmatrix}^T,$$

$$C = \begin{bmatrix} 1.5119 & 0 & 2 & 0 & 1.5119 & 0 \\ 1.3093 & 0 & 0 & 0 & -1.3093 & 0 \end{bmatrix}, \quad D = 0,$$

$$E = \text{diag}(.00049, .00599, .00073, .00322, .00048, .00151),$$

and $F = \text{diag}(.05277, .05277)$, whose output response $\{y_k\}$ is computed using random (0,1) normal distribution of input, process noise and output noise trajectories $\{u_k\}$, $\{w_k\}$, and $\{v_k\}$, $k = 1\colon 120$ ($t = 120$), and zero initial state. The parameter s is taken as 8.

The computed singular values, rounded to five significant digits, are: 3.6458, 1.7754, .75446,

.68453, .44332, .34115, .32113, .28318, .25693, .24346, .19697, .17021, .12216, .094313, .091442, .076357. These values indicate that the system order can be assumed as two. However, we forced the value $n = 6$, as for the given system. The estimated system matrices (rounded to two digits after the decimal point) are:

$$\widehat{A} = \begin{bmatrix} .96 & .38 & .08 & .17 & -.20 & .07 \\ -.10 & .96 & .62 & -.04 & -.10 & .05 \\ .02 & -.11 & .91 & .43 & -.22 & .25 \\ .01 & .06 & -.29 & .89 & .16 & .17 \\ .01 & .00 & -.06 & -.05 & -.83 & .52 \\ -.01 & -.01 & .02 & .10 & -.43 & -.64 \end{bmatrix},$$

$$\widehat{B} = \begin{bmatrix} .08 & .05 & .02 & .02 & .00 & .00 \end{bmatrix}^T,$$

$$\widehat{C} = \begin{bmatrix} .27 & -.46 & .32 & -.03 & .06 & -.12 \\ .20 & -.15 & .04 & -.49 & .41 & .06 \end{bmatrix},$$

and $\widehat{D} = 0$. The estimated Kalman gain matrix $\widehat{K}$ is

$$\widehat{K} = \begin{bmatrix} .9152 & 1.7073 \\ -.3287 & .5212 \\ .0628 & -.1079 \\ .0727 & -.0860 \\ -.0040 & -.2926 \\ -.0197 & .1059 \end{bmatrix}.$$

The relative output root-mean-square error is about 1.9, for each column of y. The same statistics for the filtered output error, for each column of y, is 0.46, and 0.58, respectively. These values are not surprising, taking into account the process and output noise considered in the system. If the noise is suppressed, that is $w_k \equiv 0$, and $v_k \equiv 0$, then the relative output root-mean-square error for the two outputs has values about 2.5×10^{-14}, and 1.8×10^{-14}, respectively, and the identified model is:

$$\widehat{A} = \begin{bmatrix} .99 & -.32 & .11 & -.25 & -.07 & .33 \\ .15 & .91 & -.57 & .09 & .03 & -.03 \\ .01 & .21 & .89 & -.37 & -.15 & .30 \\ .02 & -.00 & .14 & .92 & .71 & -.51 \\ -.00 & -.00 & .00 & -.08 & .82 & -1.43 \\ -.00 & -.00 & .00 & -.01 & .17 & .87 \end{bmatrix},$$

$$\widehat{B} = \begin{bmatrix} -.02 & .01 & -.02 & .02 & .01 & .00 \end{bmatrix}^T,$$

$$\widehat{C} = \begin{bmatrix} -.16 & -.48 & -.28 & -.04 & -.13 & -.08 \\ -.17 & -.06 & -.17 & -.51 & .56 & .47 \end{bmatrix},$$

and $\widehat{D} = 0$. The full precision computed model is close (within a small multiple of the machine precision) to a model derived from the original system by an exact similarity transformation.

5. Conclusions

Basic algorithms and LAPACK-based Fortran software for multivariable discrete-time system identification by subspace techniques, have been briefly described. The approaches seem very promising, and the software components proved robust, flexible, and easy to use, even when the available memory is quite limited. The software has been extensively tested; the results of using various algorithms (including their MATLAB versions) have been compared. New versions of the codes, with improved modularity and efficiency, are under development. Implementation of the new algorithms is also considered.

REFERENCES

1. LJUNG, L., **System Identification Toolbox**, THE MATHWORKS, INC., Natick, MA, 1992.

2. VAN OVERSCHEE, P. and DE MOOR, B., **N4SID: Two Subspace Algorithms for the Identification of Combined Deterministic-stochastic Systems**, AUTOMATICA, Vol.30, No.1, 1994, pp.75–93.

3. VERHAEGEN, M., **Identification of the Deterministic Part of MIMO State Space Models Given in Innovations Form from Input-output Data**, AUTOMATICA, Vol.30, No.1, 1994, pp.61–74.

4. MOONEN, M., DE MOOR, B., VANDENBERGHE, L. and VANDEWALLE, J., **On- and Off-line Identification of Linear State-space Models**, INT. J. CONTROL, Vol.49, No.1, 1989, pp.219–232.

5. VERHAEGEN, M. and DEWILDE, P., **Subspace Model Identification. Part 1: The Output-error State-space Model Identification Class of Algorithms**, INT. J. CONTROL, Vol.56, No.5, 1992, pp.1187–1210.

6. VERHAEGEN, M., **Subspace Model Identification. Part 3: Analysis of the Ordinary Output-error State-space Model Identification Algorithm**, INT. J. CONTROL, Vol.58, No.3, 1993, pp.555–586.

7. VAN OVERSCHEE, P., **Subspace Identification : Theory – Implementation – Applications**, Katholieke Universiteit Leuven, Ph. D. Thesis, 1995.

8. VERHAEGEN, M., **State Space Model Identification Toolbox**, Delft University of Technology, Technical Report, November 1993.

9. ANDERSON, E., BAI, Z., BISCHOF, C., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., OSTROUCHOV, S. and SORENSEN, D., **LAPACK Users' Guide: Second Edition**, SIAM, Philadelphia, PA, 1995.

10. CHO, Y. M., XU, G. and KAILATH, T., **Fast Recursive Identification of State Space Models via Exploitation of Displacement Structure**, AUTOMATICA, Vol.30, No.1, 1994, pp.45–59.

11. VAN HUFFEL, S. and VANDEWALLE, J., **The Total Least Squares Problem: Computational Aspects and Analysis**, SIAM, Philadelphia, PA, 1991.

12. SIMA, V., **Algorithms for Linear-quadratic Optimization**, in E. J. Taft and Z. Nashed (Eds.) Volume 200 of "Pure and Applied Mathematics: A Series of Monographs and Textbooks", MARCEL DEKKER, Inc., New York, 1996.

13. SIMA, V., **Algorithms and LAPACK-based Software for Subspace Identification**, Proceedings of the IEEE International Symposium on Computer-Aided Control System Design—CACSD'96, Dearborn, MI, September 15–18, 1996, U.S.A. (to appear).